Contributor: *Luka Anicin*
Data Scientist: *Ruslan S.*

CAPSTONE FINAL REPORT: Sentiment Analyses

PROBLEM: Predict the image of the company based on user's reviews (in our case it's tweets).

What are the possible ways for corporations (small or big) to measure and understand their image status (representation) among their users (customers or followers), with approximately result in accuracy of 65-89%, by avoiding expensive, biased, and time-consuming in-person surveys, through the method of analyzing the 'language' data (reviews or tweets) by the end of October 2021.

APPROACH: Wrangling, EDA, Pre-processing, Modeling, Evaluating, Tuning, Saving.

A significant part of this project will be spent on careful data pre-processing. It includes but is not limited to retrieving related to the task data from trusted sources (like Kaggle or Tweeter websites). Then, clean it (using the wrangling process) so we can apply 'special' libraries (tools) to parse and remove noise accordingly. Next, we will gather insights that could be well applied to our models. Because we know, if data is well processed, usually it boosts the accuracy of prediction by a significant amount. Next, we split data into training and testing parts so we can use the training part to train our model and testing one to evaluate it. We will be using CNN architecture models of neural networks for dealing with sequential data (1 dimensional). Once the models are training and evaluated, we will choose the most promising one based on our requirements for further production stages (deployment part).

DATA USED: Data was taken from Kaggle.com

Due to the limited resources, the data was taken from Kaggle.com

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1600000 entries, 0 to 1599999
Data columns (total 6 columns):
 #   Column  Non-Null Count    Dtype
---  ------  --------------    -----
 0   0       1600000 non-null  int64
 1   1       1600000 non-null  int64
 2   2       1600000 non-null  object
 3   3       1600000 non-null  object
 4   4       1600000 non-null  object
 5   5       1600000 non-null  object
dtypes: int64(2), object(4)
memory usage: 73.2+ MB
```

- Size of the data file is 227 MB (238,803,811 bytes)
- Shape is (1600000, 6)

DATA TIMEFRAME: 06/04/2009-29/05/2009

The data we were using for this project was pretty much outdated. So, if we are planning to build the model for today's production, then it will be a good practice to find more relevant data (new one).

MODEL REPONSE: Binary 0 or 1. Negative = 0, Positive = 1.

The data we were using had a pre-defined class. So, we were building the model that produces the answer in terms of two classes.

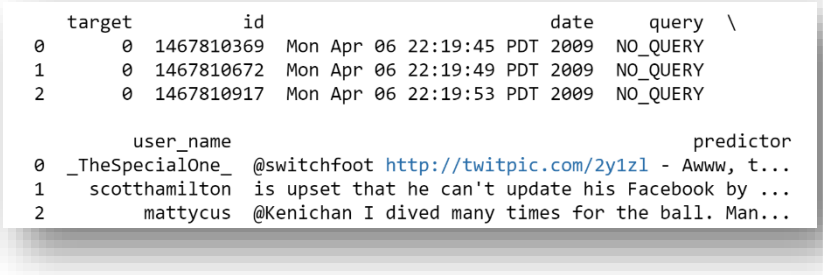
TYPE OF MODEL: Supervised Classification using Deep Learning architecture.

This project was focused on building the model using the Deep Learning approach. The most significant factor for choosing CNN was the amount of data we are going to use to train our model.

DELIVERABLES WILL BE GENERATED: PDF key points from data exploration to best model results.

DATA PRE-PROCESSING STEPS: Wrangling.

- Loading (examining the type of columns and renaming them accordingly):
    - Column names: 'target', 'id', 'date', 'query', 'user_name', 'predictor'



	target	id	date	query	\
0	0	1467810369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	
1	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	
2	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	

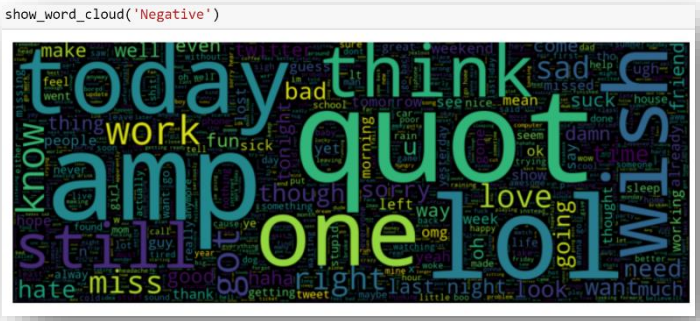
	user_name	predictor
0	_TheSpecialOne_	@switchfoot http://twitpic.com/2y1zl - Awww, t...
1	scotthamilton	is upset that he can't update his Facebook by ...
2	mattycus	@Kenichan I dived many times for the ball. Man...

- Dropping (identifying dependent and independent variables, irrelevant are dropped):
    - Dropped columns: 'id', 'date', 'query', 'user_name'

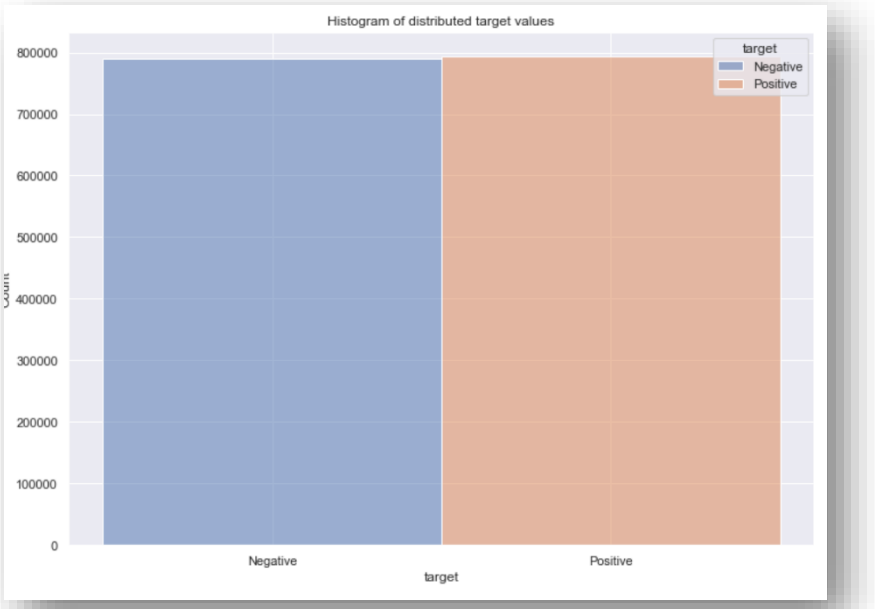
- Dealing with missing values (rows/columns that had to many missing values are dropped):
    - There were no missing values
- Dealing with duplicates (the same rows are dropped):
    - Dropped 16,309 rows
- Dealing with balanced/imbalanced data:
    - Fortunately our data were mostly balanced, no actions were applied
- Dealing with stop words (words that have not much meaning is removed):
    - We were using NLTK(Natural Language Toolkit) in python has a list of stop words. Link to the source is provided later.
- Applying regular expression function (removing the noise in the text of irrelevant characters/symbols):
    - Regular expression patter to remove noise from text: "@\S+|https?:\S+|http?:\S|[^A-Za-z0-9]+"
- Applying lemmatization (converting words in its root/based form for the further analyses):
    - For this task was used SnowballStemmer('english') package
- Splitting data (the data is split into training and testing datasets):
    - Train size is 1,266,952 by 2
    - Test size is 316,739 by 2

EDA: Visualization part to get insights.

- Exploring ration/frequencies of positive and negative words:
    - As we can see some words are unique. They can be measured as positive and as negative once. So, in this scenario, they depend on the context of the sentence.



- Checking if our model has balanced data/classes:
    - As we can see, we only have two classes, and based on the graph they are pretty much balanced out. No additional actions on the data are required.



FEATURES: There is only one column that will be used as independent feature to train model.

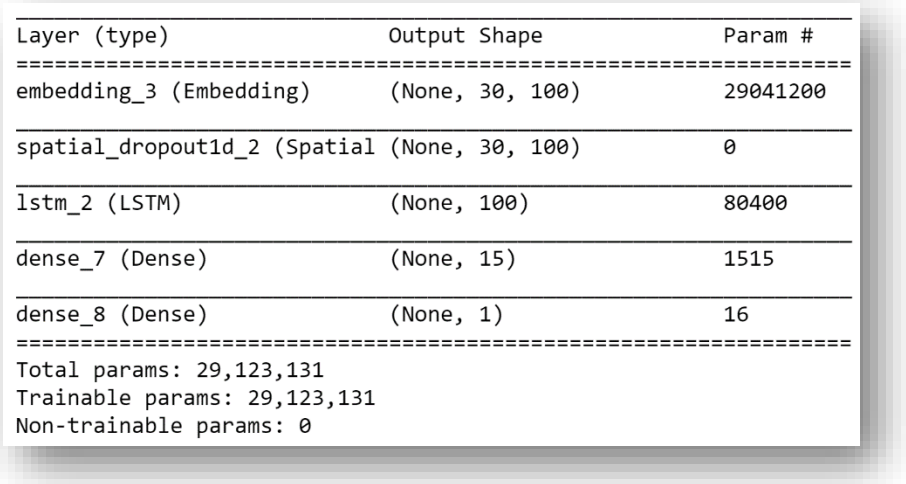
- Feature column name: 'predictor'

	target	predictor
587328	Negative	tried still work suggestions
433711	Negative	photo idea also back amp really itchy
1058707	Positive	fllw maritza 6121 hey fllwrs wl try thank ya m...
211457	Negative	pity djokovic
1414244	Positive	head 7 hours driving group people numerous win...

MODEL DESCRIPTION: Architecture and compilation parts.

- Model is 'Sequential' (CNN convolutional 1 dimension) that consists of 5 hidden layers
- Embedding layer:
    - Input data size (length) is 290412
    - Maximum sequence length is 30
    - Embedding dimension is 100
- SpatialDropout1D layer:
    - Fraction of the input units to drop is 0.2
- LSTM (long-sort term memory) layer:
    - Units (positive integer, dimensionality of the output space) is 100

    - Dropout (fraction of the units to drop for the linear transformation of the inputs) is 0.2
    - Recurrent dropout (fraction of the units to drop for the linear transformation of the recurrent state) is 0.2
- Dense layer:
    - Units (positive integer, dimensionality of the output space) is 15
    - Activation (activation function to use) is 'relu' (the rectified linear activation function)
- Dense layer:
    - Units (positive integer, dimensionality of the output space) is 1
    - Activation (activation function to use) is 'sigmoid (the rectified linear activation function)
- Model's compiler:
    - Optimizer (name of optimizer) is 'adam'
    - Loss (name of loss function) is 'binary_crossentropy'
    - Metrics (list of metrics to be evaluated by the model during training and testing) is 'accuracy'
- Epoch size is 10
- Batch size is 512
- Total parameters: 29,123,131
- Trainable parameters: 29,123,131
- Non-trainable parameters: 0



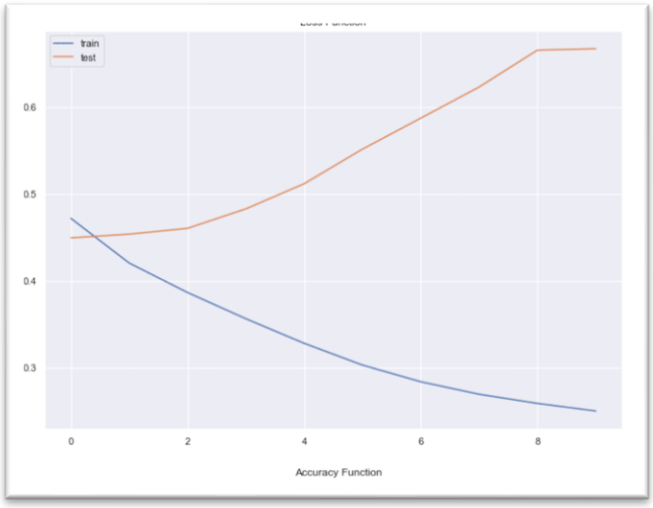
Layer (type)	Output Shape	Param #
embedding_3 (Embedding)	(None, 30, 100)	29041200
spatial_dropout1d_2 (Spatial	(None, 30, 100)	0
lstm_2 (LSTM)	(None, 100)	80400
dense_7 (Dense)	(None, 15)	1515
dense_8 (Dense)	(None, 1)	16

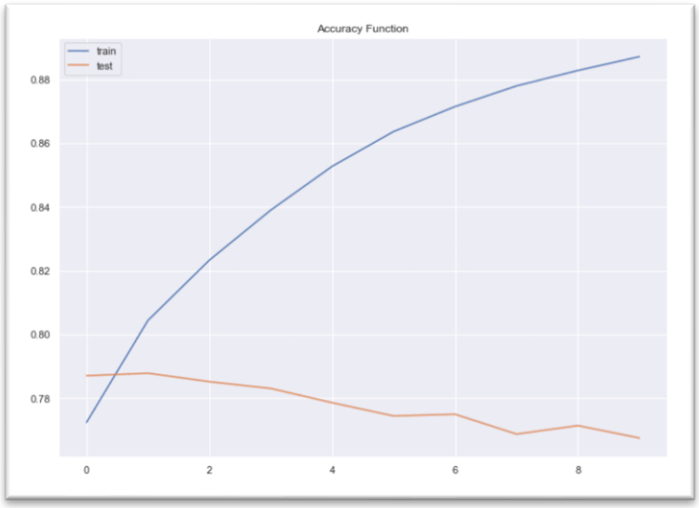
Total params: 29,123,131
Trainable params: 29,123,131
Non-trainable params: 0

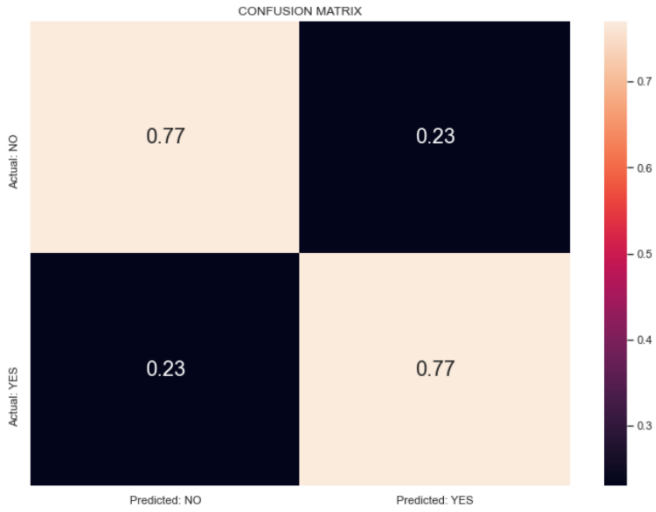
MODEL PERFORMANCE (chosen/final model number 4):

- Model name: 'model_4'
- Model performance on average:
    - Accuracy is 77%
    - Precision (macro/weighted average) is 77%

    - Recall (macro/weighted average) is 77%
    - F1-score (macro/weighted average) is 77%
- Class 0 (negatives)
    - Precision is 77%
    - Recall is 77%
    - F1-score is 77%
- Class 1 (positives)
    - Precision is 77%
    - Recall is 77%
    - F1-score is 77%
- Confusion matrix:
    - Actual ‘negatives’ and predicted ‘negatives’ is 77%
    - Actual ‘positives’ and predicted ‘positives’ is 77%
    - Actual ‘positives’ but predicted ‘negatives’ is 23%
    - Actual ‘negatives’ but predicted ‘positives’ is 23%







PERFORMENS OF ALL TRAINED MODELS: Metrics of all trained models.

Model_name	Precision	Recall	F1-score	Accuracy
Model_1	0.75	0.75	0.75	0.75
Model_3	0.75	0.75	0.75	0.75
Model_4	0.77	0.77	0.77	0.77

MODEL FINDINGS: Predictor feature (tweets column) is the only column that was used to train our model. It is significant to our task.

Because our data were well balanced our model made good predictions toward true positives and true negatives.

NEXT STEPS: The model is ready for production. It also is ready for the further hyper-parameter improvements and new data fitting.

IDEAS FOR FURTHER RESEARCH: There are some steps that could be done in case to improve overall performance of this model.

First of all, we could continue tuning our final model by using different hyper-parameters. Unfortunately, it takes a significant amount of time and proper hardware resources, which I do not possess at this moment. Also, we can apply additional libraries dealing with text. For example, we can fix words that are grammatically not correct (so, we can use word embedding with pre-trained weights more effectively). Also, we can define and split words that were used as a combination of two or more for further sentiment analyses. Finally, we could play with the architecture of our model by creating more hidden layers in case to reach the desired point. But we should stay cautious by not making the model too complex and overfitted.

CREDITS: A huge thanks to Luka Anicin for being an amazing teacher and Springboard mentor.

CITATIONS:

Team, K. (n.d.). *Keras documentation: LSTM layer*. Keras. Retrieved September 21, 2021, from https://keras.io/api/layers/recurrent_layers/lstm/.

Team, K. (n.d.). *Keras documentation: Dense layer*. Keras. Retrieved September 21, 2021, from https://keras.io/api/layers/core_layers/dense/.

Tf.keras.model : tensorflow core v2.6.0. TensorFlow. (n.d.). Retrieved September 21, 2021, from https://www.tensorflow.org/api_docs/python/tf/keras/Model#compile.

Removing stop words with NLTK in python. GeeksforGeeks. (2021, May 31). Retrieved September 21, 2021, from https://www.geeksforgeeks.org/removing-stop-words-nltk-python/.